WEST

Generate Collection | Print

Search Results - Record(s) 1 through 2 of 2 returned.

☐ 1. Document ID: US 6463527 B1

L5: Entry 1 of 2 File: USPT Oct 8, 2002

DOCUMENT-IDENTIFIER: US 6463527 B1
TITLE: Spawn-join instruction set architecture for providing explicit multithreading

US Patent No. (1):
6463527

Detailed Description Text (42):
Alternatively, global variables can be used to store local variables with proper management by the compiler or even the programmer. Good static (i.e., by compiler), or dynamic, scheduling should avoid initiating too many threads. This will alleviate a later need to put threads on hold. Good scheduling should also aim not to be starved for parallelism due to lack of advancement along critical (or possibly non-critical) paths.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 5404469 A

L5: Entry 2 of 2 File: USPT Apr 4, 1995

DOCUMENT-IDENTIFIER: US 5404469 A
TITLE: Multi-threaded microprocessor architecture utilizing static interleaving

US Patent No. (1):
5404469

Brief Summary Text (9):
Many multi-threaded architectures have been proposed to achieve higher performance and improved resource utilization in a single chip microprocessor. In R. G. Prasadh and C. L. Wu, "A Benchmark Evaluation of a Multi-threaded RISC Processor Architecture," Proc. of the International Conference on Parallel Processing, 1991, a superscaler architecture based on a VLIW model is proposed to explore the performance of a multi-threaded architecture. A dynamic interleaving technique is proposed to solve the resource contention problem. In G. E. Daddis, Jr. and H. C. Tong, "The Concurrent Execution of Multiple Instruction Streams on Superscaler Processors," Proc. of the International Conference on Parallel Processing, 1991, a system is disclosed wherein there is concurrent processing of two threads on a superscaler processor and wherein an instruction dispatch stack is used to schedule instructions at runtime. A dynamic register allocation technique is utilized to exploit both intra-thread and inter-thread instruction level parallelism.

Brief Summary Text (10):
In these prior art systems, dynamic interleaving and scheduling techniques are used

to solve the contention of resources among threads.

Detailed Description Text (5):
FIG. 4B schematically illustrates how the allocation of hardware resources specified by the allocation vector is exposed to the post-pass parallel compiler to schedule the instructions of each thread. FIG. 4B shows a list of horizontal instruction words with numbers 1,2,3, ... Each horizontal instruction word comprises four sections, with one section corresponding to each function unit. In accordance with the present invention, the parallel compiler can fill only the shaded sections with machine instructions to be executed by the corresponding function units. Unshaded sections receive NOOP instructions. The pattern of horizontal instruction words with particular shaded sections repeats itself every T instruction words where T is the number of threads. As is discussed in greater detail below, each pattern of instructions, which in the example of FIG. 4B, comprises four HIWs, is compacted into a single horizontal instruction word by the parallel compiler. The same pattern of horizontal instruction words is utilized by the compiler to individually schedule the instructions of each thread.

Detailed Description Text (7):
In other words, the resource constraints, which are imposed by the resource allocation strategy used in the hardware, are represented as a sequence of instruction patterns in the parallel compiler. Each thread is first sequentially compiled and the generated sequential code is scheduled in horizontal instruction words according to this pattern. In the parallel compiler, instructions for each thread generated by the sequential compiler, are scheduled in the pattern in a manner which preserves data dependencies and control dependencies among the instructions. The set of horizontal instruction words making up each pattern is then compacted to form a single horizontal instruction word. The compaction step is discussed in greater detail below.

Generate Collection | Print

Term	Documents
SCHEDUL$	0
SCHEDUL	8
SCHEDULA	1
SCHEDULABILITY	39
SCHEDULABLE	187
SCHEDULABLE-THIS	1
SCHEDULABLY	1
SCHEDULAGE	1
SCHEDULAING	1
SCHEDULAR	67
SCHEDULATITY	1
(L3 AND (SCHEDUL$ WITH THREAD$)).USPT.	2

There are more results than shown above. Click here to view the entire set.

WEST

Generate Collection | Print

Search Results - Record(s) 157 through 157 of 157 returned.

☐ 157. Document ID: US 3736566 A

L15: Entry 157 of 157 File: USPT May 29, 1973

DOCUMENT-IDENTIFIER: US 3736566 A
TITLE: CENTRAL PROCESSING UNIT WITH HARDWARE CONTROLLED CHECKPOINT AND RETRY FACILITIES

Detailed Description Text (63):
There has thus been shown in one form of the present invention means for creating a precise data processing system condition. Processing proceeds with the execution of a sequence of program instructions while saving the original contents of only those data registers which are modified during the processing. The invention provides the ability to return the data processing system to the previously established precise state by restoring the contents of data registers which have been modified and return of the data processing system control state to the condition that existed at the time of establishing the precisely known state. In response to either manually or programmed control signals, the previous sequence of instructions can be retried. The retry of the instruction sequence can be on an individual instruction basis, that is out of overlap, or can proceed in an overlap fashion up to a particular point at which time instructions will be executed out of overlap. Further, once recovery to the previous state has been reached, the data processing system may initiate an entirely different instruction sequence in dependence on the condition which caused return to the previously established checkpoint. The retry of a particular instruction sequence in a non-overlapped mode of operation permits a determination to be made of the precise cause of an interrupt or hardware error condition.

Current US Cross Reference Classification (1):
712/228

Full | Title

Generate Collection | Print

Term	Documents
STATE	1003399
STATES	351003
INTERRUP$	0
INTERRUP	92
INTERRUPATED	1
INTERRUPATION	1
INTERRUPCHARACTER	1
INTERRUPE	2
INTERRUPED	174
INTERRUPER	19
INTERRUPERS	2
(L13 AND (INTERRUP$ SAME SAV$ SAME RETURN$ SAME STATE)).USPT.	157

There are more results than shown above. Click here to view the entire set.

Display Format: KWIC Change Format

Previous Page Next Page

L7: Entry 2 of 3 File: USPT Apr 2, 2002

DOCUMENT-IDENTIFIER: US 6366998 B1
**** See image for Certificate of Correction ****
TITLE: Reconfigurable functional units for implementing a hybrid VLIW-SIMD programming model

US Patent No. (1):
6366998

Brief Summary Text (5):
Instructions are processed by a scheduler which determines which functional units should be used for executing each instruction. Scheduling may be done statically, i.e., at compile time, as opposed to dynamically, i.e., at run time. Thus, VLIW models can simultaneously execute instructions while minimizing the occurrence of hazards. Because of this feature, among others, instruction parallelism models are very efficient in telecommunications applications.

Brief Summary Text (6):
Developing an instruction set architecture based on a VLIW model has several advantages. First, VLIW models are very scalable, both upward and downward. Scalability refers to the number of operations that can be packed into one long instruction word. The scalability enables the model to serve as a basis for a family of derivative implementations for various high performance digital signal processor ("DSP") and multimedia applications. Second, "memory walls" are not an issue in the VLIW model. Memory walls refer to the concept that processor speeds are increasing at a rate more quickly than memory speeds. In the case of a VLIW model, memory walls are not a concern because the processor is simultaneously executing a large number of instructions instead of executing one complex instruction in a consecutive order where a processor would have to repeatedly wait for information from memory for every consecutive instruction. Third, the VLIW model saves silicon area and power by off loading the complex instruction scheduling scheme to the compiler.

Brief Summary Text (13):
Accordingly, the present invention overcomes problems in the prior art by providing an instruction set architecture for a digital signal processor that has improved code density, improved instruction level parallelism and improved issue bandwidth. The instruction set architecture includes information packets which may include instructions having different characteristics, such as instruction type (for example, scalar or vector) and instruction length (for example, 16-bit and 32-bit). These instructions are received by a scheduler or scoreboard unit which then determines the functional units that are available for executing the instructions. The instructions are then broadcast to a plurality of function units or processing elements for execution.

Drawing Description Text (13):
FIG. 11 is a schematic representation of an exemplary scheduler.

Detailed Description Text (6):
Instructions 104-116 of the hybrid VLIW-SJMD DSP 100 are preferably received by a scheduler or scoreboard unit 120 which then determines which functional units are available for executing the instructions. Instructions 104-116 are then broadcast data path units ("DPUs") 122, each of which typically includes a plurality of functional units or processing elements. An exemplary DSP 100 preferably includes five DPUs 122. The functional units or processing elements included within DPUs 122 execute instructions 104-116 utilizing data element or operands from a scalar register file 124 and a vector register file 126.

Detailed Description Text (15):
Mode bits are typically located in a multiple-bit template field identifying specifications of an instruction packet which is contained in each instruction packet in accordance with a preferred embodiment of the present invention. In addition to a mode bit sub-field, a template field may also include the following sub-fields: a grouping field (which contains instruction issue groups), a thread identifier, and a repeat field (which identifies whether the entire instruction packet is to be repeated as a zero-overhead loop).

Detailed Description Text (17):
Related to the issue group is the issue bandwidth, which represents the number of simple instructions that can be issued, i.e., physically dispatched to execution units, per second. If the issue bandwidth is much smaller than the fetch bandwidth, i.e., the number of VLIW fetch packets that a DSP can fetch per second, the performance of the processor will deteriorate significantly. In other words, a DSP will not be operating efficiently if it is fetching instructions faster than it is executing the instructions and creating a buildup or backlog of instructions. This may be a result of a largely serial or scalar application, which does not take advantage of the parallel resources provided by the processor. This may also be the result of poor instruction scheduling in that the scheduler is not searching broadly enough for independent instructions that can be issued out of order and that can utilize the issue bandwidth of the processor. To achieve more efficient processing, VLIW fetch packets are preferably filled to capacity to take advantage of code density and issue groups are preferably maximized within each fetch packet.

Detailed Description Text (24):
The configurability of the present hybrid VLIW-SIMD DSP may require intelligence in the hardware to execute the operations in the instruction packets. In general, instruction packets are broadcast to a plurality of processing elements or functional units where each instruction packet contains instructions of various characteristics as discussed above. These characteristics may include, inter alia, varying instruction types and varying instruction lengths. The instruction packet need not identify which specific functional units should be used in executing the various types of instructions. Rather, a scheduler in a DSP is preferably designed to schedule instructions for particular functional units depending on the specific instructions. In a subsequent cycle, the scheduler may reconfigure the coupling or grouping of the functional units and schedule different instructions to them for execution. The reconfiguration ability reduces the amount of execution time needed and reduces the chance of hazards, such as read after write ("RAW") hazards. The functional unit configurability is preferably facilitated by buses feeding source operands to the functional units, buses transferring the results, and the scheduling logic for implementing result forwarding and bypass paths.

Detailed Description Text (32):
The hardware may determine which of the functional units are available to perform the operations specified in the instruction packets via a hardware scheduler, reconfiguring element or scoreboard unit which tracks which functional units are currently performing operations and which units are available to perform operations. The scheduler determines functional unit availability in a way such that the comparators are minimized and the cycle time is not increased. With reference to FIG. 1, an exemplary scheduler 1100 is shown charting destination operands 1102 against source operands 1104. For example, if a particular ALU is currently performing an operation, scheduler 1100 identifies what other ALUs are available to receive data to perform an operation and which ALUs cannot receive the data. Scheduler 1100 is configured based upon how the DPUS and other processing elements are associated and connected.

CLAIMS:

15. The processor of claim 14, wherein said reconfiguring element is configured to receive said first instructions and said second instructions, to determine the availability of said functional units and to schedule said first instructions and said second instructions to said functional units for execution.

18. The digital signal processing system of claim 17, wherein said reconfiguring element receives said first instructions, said second instructions, and said third instructions, to said functional units and wherein said functional units execute said scheduled instruction.

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L9	L8 and (network$ or internet)	0	L9
L8	L7 and packet$	1	L8
L7	L3 and (schedul$ and thread$)	3	L7
L6	L3 and (schedul$ same thread$)	2	L6
L5	L3 and (schedul$ with thread$)	2	L5
L4	L3 and (schedul$ with first with second with thread$)	0	L4
L3	(6366998 or 6311261 or 5404469 or 5768528 or 5933627 or 6167503 or 6463527).pn.	7	L3
L2	L1 and (schedul$ with thread$)	0	L2
L1	6373848.pn.	1	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: US 6366998 B1

L8: Entry 1 of 1 File: USPT Apr 2, 2002

DOCUMENT-IDENTIFIER: US 6366998 B1
**** See image for Certificate of Correction ****
TITLE: Reconfigurable functional units for implementing a hybrid VLIW-SIMD programming model

Abstract Text (1):
The present invention generally relates to a hybrid VLIW-SIMD programming model for a digital signal processor. The hybrid programming model broadcasts a packet of information to a plurality of functional units or processing elements. Each packet contains several instructions having certain characteristics, such as instruction type and instruction length, among others. The hybrid programming model includes functional units which are reconfigurable based upon the instructions with an instruction packet and the availability of the functional units. The model groups the functional units such that the operations specified in the instructions can be efficiently executed and selects which functional units should be utilized for a given operation.

US Patent No. (1):
6366998

Brief Summary Text (5):
Instructions are processed by a scheduler which determines which functional units should be used for executing each instruction. Scheduling may be done statically, i.e., at compile time, as opposed to dynamically, i.e., at run time. Thus, VLIW models can simultaneously execute instructions while minimizing the occurrence of hazards. Because of this feature, among others, instruction parallelism models are very efficient in telecommunications applications.

Brief Summary Text (6):
Developing an instruction set architecture based on a VLIW model has several advantages. First, VLIW models are very scalable, both upward and downward. Scalability refers to the number of operations that can be packed into one long instruction word. The scalability enables the model to serve as a basis for a family of derivative implementations for various high performance digital signal processor ("DSP") and multimedia applications. Second, "memory walls" are not an issue in the VLIW model. Memory walls refer to the concept that processor speeds are increasing at a rate more quickly than memory speeds. In the case of a VLIW model, memory walls are not a concern because the processor is simultaneously executing a large number of instructions instead of executing one complex instruction in a consecutive order where a processor would have to repeatedly wait for information from memory for every consecutive instruction. Third, the VLIW model saves silicon area and power by off loading the complex instruction scheduling scheme to the compiler.

Brief Summary Text (13):
Accordingly, the present invention overcomes problems in the prior art by providing an instruction set architecture for a digital signal processor that has improved code density, improved instruction level parallelism and improved issue bandwidth. The instruction set architecture includes information packets which may include instructions having different characteristics, such as instruction type (for

example, scalar or vector) and instruction length (for example, 16-bit and 32-bit). These instructions are received by a scheduler or scoreboard unit which then determines the functional units that are available for executing the instructions. The instructions are then broadcast to a plurality of function units or processing elements for execution.

Drawing Description Text (5):
FIG. 3 is a schematic representation of an exemplary instruction packet having all scalar 32-bit instructions;

Drawing Description Text (6):
FIG. 4 is a schematic representation of an exemplary instruction packet having all scalar 16-bit instructions;

Drawing Description Text (7):
FIG. 5 is a schematic representation of an exemplary instruction packet having scalar and vector 32-bit instructions intermixed;

Drawing Description Text (8):
FIG. 6 is a schematic representation of an exemplary instruction packet having scalar and vector 16-bit instructions intermixed;

Drawing Description Text (9):
FIG. 7 is a schematic representation of an exemplary instruction packet having scalar and vector 16-bit and 32-bit instructions intermixed;

Drawing Description Text (13):
FIG. 11 is a schematic representation of an exemplary scheduler.

Detailed Description Text (5):
With reference to FIG. 1, an exemplary hybrid VLIW-SIMD DSP 100 executes a number of 256-bit instruction packets 102 (one shown in FIG. 1) which include, among other things, a number of instructions 104-116, and a header or template field 118. Instructions 104-116 specify operations to be performed on specified operands and may be of varying characteristics, such as instruction length and instruction type. Instructions 104-116 within instruction packet 102 may be of different lengths, for example, all 16 bits long, all 32 bits long, or a combination of lengths. The length characteristics of instructions 104-116 may be identified in template field 118. In addition, instructions 104-116 may be of different types, for example, all scalar instructions, all vector instructions or some combination of instructions types. It should be appreciated that the above bit lengths, characteristics, instruction types and instruction lengths are exemplary and shall not be so limited. These exemplary characteristics may extend to include other characteristics currently known or those that may be utilized in the future.

Detailed Description Text (6):
Instructions 104-116 of the hybrid VLIW-SJMD DSP 100 are preferably received by a scheduler or scoreboard unit 120 which then determines which functional units are available for executing the instructions. Instructions 104-116 are then broadcast data path units ("DPUs") 122, each of which typically includes a plurality of functional units or processing elements. An exemplary DSP 100 preferably includes five DPUs 122. The functional units or processing elements included within DPUs 122 execute instructions 104-116 utilizing data element or operands from a scalar register file 124 and a vector register file 126.

Detailed Description Text (8):
Put another way, if a programmer desires to develop a program using only the horizontal instruction types to satisfy an entirely scalar application, he would be able to take advantage of the multiple functional units because there would be no vectorizable instructions utilizing the functional units. That is, each instruction in a VLIW instruction packet can simultaneously utilize different functional units in a more time-efficient manner. As here, when a program only uses a VLIW mode and does not contain any vectorizable instructions, it is referred to as horizontal parallelism.

Detailed Description Text (10):
In some circumstances, especially if maximum efficient performance is desired, the hybrid VLIW-SIMD model may be exploited to get both horizontal and vertical parallelism. For instance consider the case, where a VLIW packet contains the above VADD in parallel with a scalar load, a scalar multiply and a vector subtract. In this exemplary situation, we have exploited the hybrid model and taken advantage of both the horizontal parallelism (represented by scalar instructions within a VLIW packet) and the vertical parallelism (represented by vector instructions that perform the same operation on several elements). The above situation is depicted by the following instruction, where ".parallel." means in parallel with.

Detailed Description Text (12):
With reference now to FIG. 2, an alternate embodiment of an exemplary hybrid VLIW-SIMD DSP 200 executes a number of 128-bit instruction packets 202. The characteristics of instructions within instruction packet 202 may be varied as described above with reference to FIG. 1. In this alternative embodiment, a DSP of model 200 includes a fewer number of DPUd 206, preferably three, although other numbers could be used, because the instructions in instruction packet 202 are smaller and need fewer functional units in DPUd 206 to execute the instructions. Functional units within DPUs 206 use operands specified in the instructions in instruction packet 202 from scalar register file 210 and/or vector register file 208.

Detailed Description Text (13):
As stated above, the ISA of the hybrid VLIW-SIMD programming model provides for instruction packets having different bit lengths. In accordance with conventional techniques, the instruction packets may have a bit length equal to an integer multiple of eight. Preferably, instruction packets are 256 bits in length as shown in FIG. 1, but this length is exemplary and not so limited. The instructions within the instruction packets may be of various lengths, preferably 32-bit instructions (i.e., long format) and 16-bit instructions (i.e., short format). The dual instruction length set of the present programming model strikes a balance between the improved code density, improved instruction level parallelism, and improved issue bandwidth and efficiency of a fixed-length standard 32-bit reduced instruction set computer ("RISC"). The dual instruction set improves code density in that if an instruction is able to be written using 16-bit instructions, it can be written without wasting the additional memory and time requirements associated with 32-bit instructions. In addition, the dual instruction set improves instruction level parallelism in that now more instructions can be included in an instruction packet. Without the dual instruction set, the maximum number of purely scalar instructions a fixed-length 32-bit instruction set of a 256-bit instruction packet is seven. However, with the inclusion of 16-bit instructions, the maximum number of purely scalar instructions increases to fourteen. The dual instruction set also improves the issue bandwidth, as well as the instruction level parallelism, because the instructions issued per cycle may significantly increase.

Detailed Description Text (14):
In a preferred embodiment, one or more mode bits identify which instruction format is used for each instruction in an instruction packet. As noted above, the hybrid VLIW-SIMD programming model enables the intermixing of 16-bit and 32-bit instructions within one instruction packet, and in the event that the intermixing does occur, the total number of consecutive 16-bit instruction sets in an instruction packet should be even. Preferably, each instruction packet contains a number of mode bits to identify whether each instruction contained in the packet is in the long or short format. A mode field may contain a number of mode bits. For example, a mode field of a 256-bit instruction packet is preferably a 14-bit field containing seven sub-fields, each two bits wide. A first bit value can be set to identify a long format instruction, and a second bit value can be set to identify a short format instruction. Each sub-field corresponds to one long format instruction or two short format instructions. Mode bits may be set and cleared by a move-to-control register instruction. In other words, a given program can use short format instructions, and, after a mode bit is cleared, use long format instructions.

Detailed Description Text (15):

Mode bits are typically located in a multiple-bit template field identifying specifications of an instruction packet which is contained in each instruction packet in accordance with a preferred embodiment of the present invention. In addition to a mode bit sub-field, a template field may also include the following sub-fields: a grouping field (which contains instruction issue groups), a thread identifier, and a repeat field (which identifies whether the entire instruction packet is to be repeated as a zero-overhead loop).

Detailed Description Text (16):
A grouping field may include a fetch packet which typically includes three issue groups, i.e., the number of independent instructions that can be issued concurrently in the same cycle. Instruction packets may be fetched from a storage location during a fetch cycle, i.e., that portion of an instruction cycle during which a fetch occurs, as opposed to the instruction execution portion and the like. During a fetch cycle, a fetch packet, which may be a fixed length VLIW packet, is fetched or retrieved from memory. In an exemplary embodiment of the present invention, a fetch packet is an entire 256-bit VLIW instruction packet having at least seven instruction slots and a control template specifying the issue groups within the packet. A last issue group in a VLIW fetch packet can chain instructions with a subsequent VLIW fetch packet.

Detailed Description Text (17):
Related to the issue group is the issue bandwidth, which represents the number of simple instructions that can be issued, i.e., physically dispatched to execution units, per second. If the issue bandwidth is much smaller than the fetch bandwidth, i.e., the number of VLIW fetch packets that a DSP can fetch per second, the performance of the processor will deteriorate significantly. In other words, a DSP will not be operating efficiently if it is fetching instructions faster than it is executing the instructions and creating a buildup or backlog of instructions. This may be a result of a largely serial or scalar application, which does not take advantage of the parallel resources provided by the processor. This may also be the result of poor instruction scheduling in that the scheduler is not searching broadly enough for independent instructions that can be issued out of order and that can utilize the issue bandwidth of the processor. To achieve more efficient processing, VLIW fetch packets are preferably filled to capacity to take advantage of code density and issue groups are preferably maximized within each fetch packet.

Detailed Description Text (18):
To illustrate the interaction of the hybrid VLIW-SIMD programming model having differing instructions types and the differing instructions lengths set, several combinations of instructions in an instruction packet can be executed. Referring now to FIG. 3, an exemplary 256-bit instruction packet 300 can include seven 32-bit scalar instructions. This instruction combination would be executed in a pure VLIW mode in the long instruction format. Instruction packet 300 includes two add instructions 302 & 304, one MAC instruction 306, one load word instruction 308, one logical shift right instruction 310, one store word instruction 312 and one arithmetic shift right instruction 313, where all of the instructions are 32 bits wide. In addition, instruction packet 300 may also include a template field 314.

Detailed Description Text (19):
Referring now to FIG. 4, an exemplary instruction packet 400 could include fourteen 16-bit scalar instructions. This instruction combination would be executed in a pure VLIW mode in the short instruction format. Instruction packet 400 contains six MAC instructions 402, four add instructions 404, one load word instruction 406, one logical shift left instruction 408, one store word instruction 410, and one arithmetic shift right instruction 412, where all of the instructions are 16 bits wide. Instruction packet 400 also contains a 32-bit template field 414, as described above. If, for example, a "0" indicates a short form instruction and a "1" indicates a long instruction format, an exemplary fourteen bit mode filed in the template field would consist of all "0's." However, this convention is exemplary and other identifying conventions may also be used.

Detailed Description Text (20):
Referring now to FIG. 5, an exemplary instruction packet 500 may also have scalar and vector 32-bit instructions intermixed. Instruction packet 500 includes three

32-bit scalar instructions and four 32-bit vector instructions. More specifically, instruction packet 500 includes two vector MAC instructions 502, two vector add instructions 504, one scalar load word instruction 506, one scalar logical shift right instruction 508, and one scalar store word instruction 510, where all of the instructions are 32 bits wide. Instruction packet 500 also contains a 32-bit template field 512. Instruction packet 500 is utilizing the hybrid VLIW-SIMD feature, that is, the mixed instruction type feature, of the present invention.

Detailed Description Text (21):
Referring now to FIG. 6, an exemplary instruction packet 600 may also include scalar and vector 16-bit instructions intermixed. Instruction packet 600 includes eight 16-bit scalar instructions and six 16-bit vector instructions. More specifically, instruction packet 600 includes four vector MAC instructions 602, two vector add instructions 604, two scalar add instructions 606, one scalar load word instruction 608, one scalar arithmetic shift right instruction 610, one scalar store word instruction 612, and three scalar logical shift right instructions 614, where all of the instructions are 16 bits wide. Instruction packet 600 also contains a 32-bit template field 616.

Detailed Description Text (22):
Referring now to FIG. 7, an exemplary instruction packet 700 mixes both of the characteristics and may include scalar and vector 16-bit and 32-bit instructions intermixed. Instruction packet 700 includes three 32-bit scalar instructions, two 16-bit scalar instructions, two 32-bit vector instruction and two 16-bit vector instructions. Specifically instruction packet 700 includes two 32-bit vector MAC instructions 702, two 16-bit scalar add instructions 704, one scalar 32-bit MAC instruction 706, two 16-bit vector add instructions 708, one 32-bit scalar load word instruction 710, and one 32-bit scalar arithmetic shift right instruction 712. Instruction packet 700 also contains a 32-bit template field 714.

Detailed Description Text (23):
FIGS. 3-7 identify exemplary instruction packets. The configuration of the packets is not so limited and can vary depending upon the requirements of an application.

Detailed Description Text (24):
The configurability of the present hybrid VLIW-SIMD DSP may require intelligence in the hardware to execute the operations in the instruction packets. In general, instruction packets are broadcast to a plurality of processing elements or functional units where each instruction packet contains instructions of various characteristics as discussed above. These characteristics may include, inter alia, varying instruction types and varying instruction lengths. The instruction packet need not identify which specific functional units should be used in executing the various types of instructions. Rather, a scheduler in a DSP is preferably designed to schedule instructions for particular functional units depending on the specific instructions. In a subsequent cycle, the scheduler may reconfigure the coupling or grouping of the functional units and schedule different instructions to them for execution. The reconfiguration ability reduces the amount of execution time needed and reduces the chance of hazards, such as read after write ("RAW") hazards. The functional unit configurability is preferably facilitated by buses feeding source operands to the functional units, buses transferring the results, and the scheduling logic for implementing result forwarding and bypass paths.

Detailed Description Text (31):
The hybrid programming model embodied in a preferred embodiment of the present invention is a flexible model in that it can configure its functional units as needed. For example, a vector MAC instruction can use two MAC units in a single cycle to do a 64-bit vector MAC. On the other hand, in the VLIW mode, four scalar instructions can be issued in parallel and packed into one instruction packet to perform, for example, one MAC, one shift, one load, and one branch in a single cycle. This flexibility provides superior performance on VLIW model operations such as fast fourier transforms ("FFTs") as well SIMD model operations, such as MPEG, FIRs and image processing algorithms.

Detailed Description Text (32):
The hardware may determine which of the functional units are available to perform

the operations specified in the instruction packets via a hardware scheduler, reconfiguring element or scoreboard unit which tracks which functional units are currently performing operations and which units are available to perform operations. The scheduler determines functional unit availability in a way such that the comparators are minimized and the cycle time is not increased. With reference to FIG. 1, an exemplary scheduler 1100 is shown charting destination operands 1102 against source operands 1104. For example, if a particular ALU is currently performing an operation, scheduler 1100 identifies what other ALUs are available to receive data to perform an operation and which ALUs cannot receive the data. Scheduler 1100 is configured based upon how the DPUS and other processing elements are associated and connected.

CLAIMS:

1. An instruction set architecture for a digital signal processor comprising:

a first plurality of instruction packets, wherein each of said first plurality of instruction packets comprises a number of first instructions of a same first characteristic; and

a second plurality of instruction packets, wherein each of said second plurality of instruction packets comprises a number of second instructions different from said first characteristic, wherein said first characteristic and said second characteristic are different and said digital signal processor comprises a plurality of functional units, each functional unit reconfigurable for one of scalar and vector processing.

7. The instruction set architecture of claim 1 further comprising: a third plurality of instruction packets, wherein each of said third plurality of instruction packets comprises a number of third instructions, wherein the characteristic of each of said third instructions is selectable from said first characteristic and said second characteristic.

8. The instruction set architecture of claim 7, wherein the characteristics of at least two of said third instructions in each of said third plurality of instruction packets are different.

14. A digital signal processor for use in executing a first plurality of instruction packets, wherein each of said first plurality of instruction packets comprises a number of first instructions of a same first characteristic and for executing a second plurality of instruction packets, wherein each of said second plurality of instruction packets comprises a number of second instruction of a same second characteristic and wherein said first characteristic and said second characteristic are different and said digital signal processor comprising:

a plurality of functional units, each functional unit reconfigurable for one of scalar and vector processing;

a plurality of functional units; and

an element for reconfiguring said functional unit in accordance with said first characteristic and said second characteristic.

15. The processor of claim 14, wherein said reconfiguring element is configured to receive said first instructions and said second instructions, to determine the availability of said functional units and to schedule said first instructions and said second instructions to said functional units for execution.

17. A digital signal processor for use in executing a first plurality of instruction packets, wherein each of said first plurality of instruction packets comprises a number of first instructions of a same first characteristic, for use in executing a second plurality of instruction packets, wherein each of said second plurality of instruction packets comprises a number of second instructions of a same second characteristic and for use in executing a third plurality of instruction packets, wherein each of said third plurality of instruction packets comprises a number of

third instructions, wherein said first characteristic and said second characteristic are different and said digital signal processor comprises a plurality of functional units, each reconfigurable for one of scalar and vector processing and wherein each of said third instructions is selectable from said first characteristic and said second characteristic, said processor comprising:

a plurality of functional units;

an element for reconfiguring said functional units in accordance with said first instructions, said second instructions and said third instructions;

a scalar register file; and

a vector register file, wherein said functional units obtain information from one or more register files selectable from said scalar register file and said vector register file in accordance with said first instructions, said second instructions and said third instructions.

18. The digital signal processing system of claim 17, wherein said reconfiguring element receives said first instructions, said second instructions, and said third instructions, to said functional units and wherein said functional units execute said scheduled instruction.

19. A method of executing instructions in a digital signal processing system comprising a plurality of functional units, each reconfigurable for one of scalar and vector processing and further comprising the steps of:

receiving an instruction packet including a number of instructions, wherein a characteristic of each said instructions is selected from a plurality of characteristics;

reconfiguring a plurality of functional units to execute one of scalar and vector processing in response to each said instruction;

receiving information from a register file selectable from a scalar register file and vector register file; and

executing said instructions in said reconfigured functional units.

Applied Parallelism Hit List 2/04 Hombased

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: US 6366998 B1

L2: Entry 1 of 1 File: USPT Apr 2, 2002

DOCUMENT-IDENTIFIER: US 6366998 B1
** See image for Certificate of Correction **
TITLE: Reconfigurable functional units for implementing a hybrid VLIW-SIMD programming model

Brief Summary Text (4):
Many different types of programming models exist in the area of digital signal processing. In general, these models differ by their characteristics, such as data types, data lengths, data functions and the like. Instruction parallelism models are one type of model. An instruction parallelism model is defined by its ability to simultaneously execute different instructions. Instruction parallelism models can be embodied by a very long instruction word ("VLIW") model or a super-scalar model, among others. VLIW models use a horizontal approach to parallelism where several scalar instructions are included in a long instruction word that is fetched and executed every cycle. More specifically, in each cycle, an instruction word specifies operations to be performed using specific operands. Exemplary operations may include mathematical operations, logical operations, and the like, depending upon the needs of a particular application. Functional units which perform the operations may include any type of processing elements, such as, for example, execution units. More specifically, exemplary functional units may include multiply-accumulate ("MAC") units, load/store units, add units, etc. and may vary from application to application.

Brief Summary Text (5):
Instructions are processed by a scheduler which determines which functional units should be used for executing each instruction. Scheduling may be done statically, i.e., at compile time, as opposed to dynamically, i.e., at run time. Thus, VLIW models can simultaneously execute instructions while minimizing the occurrence of hazards. Because of this feature, among others, instruction parallelism models are very efficient in telecommunications applications.

Brief Summary Text (6):
Developing an instruction set architecture based on a VLIW model has several advantages. First, VLIW models are very scalable, both upward and downward. Scalability refers to the number of operations that can be packed into one long instruction word. The scalability enables the model to serve as a basis for a family of derivative implementations for various high performance digital signal processor ("DSP") and multimedia applications. Second, "memory walls" are not an issue in the VLIW model. Memory walls refer to the concept that processor speeds are increasing at a rate more quickly than memory speeds. In the case of a VLIW model, memory walls are not a concern because the processor is simultaneously executing a large number of instructions instead of executing one complex

h e b b g e e e f e ef b e

instruction in a consecutive order where a processor would have to repeatedly wait for information from memory for every consecutive instruction. Third, the VLIW model saves silicon area and power by off loading the complex instruction scheduling scheme to the compiler.

Brief Summary Text (7):
Data parallelism models are a second type of model. A data parallelism model, also known as a vector model, is defined by its ability to simultaneously execute multiple operations of a single instruction, where each operation can be performed with different data. A data parallelism model uses vector instructions specifying vector operations and is embodied in a single instruction multiple data ("SIMD") model. Data parallelism models are very efficient in block based applications such as image processing, Motion Pictures Experts Group ("MPEG") systems, Finite Duration Impulse Response ("FIR") systems, video conferencing, filtering applications, and multimedia applications.

Brief Summary Text (8):
As noted above, the instruction parallelism models and the data parallelism models are efficient for different types of applications. It would be extremely advantageous to develop a programming model with an instruction set architecture ("ISA") which incorporates the advantages of both the instruction parallelism and data parallelism models on which many different types of applications may run. More specifically, it would be advantageous to develop an instruction set architecture which permits the incorporation of a vertical programming model, such as a SIMD model, into a horizontal programming model, such as a VLIW model.

Brief Summary Text (13):
Accordingly, the present invention overcomes problems in the prior art by providing an instruction set architecture for a digital signal processor that has improved code density, improved instruction level parallelism and improved issue bandwidth. The instruction set architecture includes information packets which may include instructions having different characteristics, such as instruction type (for example, scalar or vector) and instruction length (for example, 16-bit and 32-bit). These instructions are received by a scheduler or scoreboard unit which then determines the functional units that are available for executing the instructions. The instructions are then broadcast to a plurality of function units or processing elements for execution.

Detailed Description Text (2):
The present invention provides a hybrid VLIW-SIMD programming model for use with a digital signal processor ("DSP"). The hybrid VLIW-SIMD programming model of the present invention is designed to facilitate efficient mapping of various telecommunication, multimedia, and DSP algorithms. The programming model is based on the concept that if multiple functional units are available, they can be configured such that they deliver both horizontal parallelism, e.g., under a VLIW programming model, vertical parallelism, e.g., under a SIMD programming model, or a combination thereof, i.e., a hybrid programming model. This model is extremely flexible in that it permits both SIMD vectorizable block-based multimedia applications as well as VLIW non-vectorizable modem and telecommunication applications.

Detailed Description Text (8):
Put another way, if a programmer desires to develop a program using only the horizontal instruction types to satisfy an entirely scalar application, he would be able to take advantage of the multiple functional units because there would be no vectorizable instructions utilizing the functional units. That is, each instruction in a VLIW instruction packet can simultaneously utilize different functional units in a more time-efficient manner. As here, when a program only uses a VLIW mode and does not contain any vectorizable instructions, it is referred to as horizontal parallelism.

Detailed Description Text (9):
Similarly, if a programmer desires to develop a program using vector (SIMD) instructions to satisfy vectorizable applications, he may be able to use multiple functional units combined to execute the instruction. For example, a vector addition instruction "VADD" performed on two vectors of 64-bits each uses two adder units combined together to perform the vector add operation. This vector execution model is referred to vertical parallelism because it is one instruction performed on multiple data elements.

Detailed Description Text (10):
In some circumstances, especially if maximum efficient performance is desired, the hybrid VLIW-SIMD model may be exploited to get both horizontal and vertical parallelism. For instance consider the case, where a VLIW packet contains the above VADD in parallel with a scalar load, a scalar multiply and a vector subtract. In this exemplary situation, we have exploited the hybrid model and taken advantage of both the horizontal parallelism (represented by scalar instructions within a VLIW packet) and the vertical parallelism (represented by vector instructions that perform the same operation on several elements). The above situation is depicted by the following instruction, where ".parallel." means in parallel with.

Detailed Description Text (13):
As stated above, the ISA of the hybrid VLIW-SIMD programming model provides for instruction packets having different bit lengths. In accordance with conventional techniques, the instruction packets may have a bit length equal to an integer multiple of eight. Preferably, instruction packets are 256 bits in length as shown in FIG. 1, but this length is exemplary and not so limited. The instructions within the instruction packets may be of various lengths, preferably 32-bit instructions (i.e., long format) and 16-bit instructions (i.e., short format). The dual instruction length set of the present programming model strikes a balance between the improved code density, improved instruction level parallelism, and improved issue bandwidth and efficiency of a fixed-length standard 32-bit reduced instruction set computer ("RISC"). The dual instruction set improves code density in that if an instruction is able to be written using 16-bit instructions, it can be written without wasting the additional memory and time requirements associated with 32-bit instructions. In addition, the dual instruction set improves instruction level parallelism in that now more instructions can be included in an instruction packet. Without the dual instruction set, the maximum number of purely scalar instructions a fixed-length 32-bit instruction set of a 256-bit instruction packet is seven. However, with the inclusion of 16-bit instructions, the maximum number of purely scalar instructions increases to fourteen. The dual instruction set also improves the issue bandwidth, as well as the instruction level parallelism, because the instructions issued per cycle may significantly increase.

Detailed Description Text (17):
Related to the issue group is the issue bandwidth, which represents the number of simple instructions that can be issued, i.e., physically dispatched to execution units, per second. If the issue bandwidth is much smaller than the fetch bandwidth, i.e., the number of VLIW fetch packets that a DSP can fetch per second, the performance of the processor will deteriorate significantly. In other words, a DSP will not be operating efficiently if it is fetching instructions faster than it is executing the instructions and creating a buildup or backlog of instructions. This may be a result of a largely serial or scalar application, which does not take advantage of the parallel resources provided by the processor. This may also be the result of poor instruction scheduling in that the scheduler is not searching broadly enough for independent instructions that can be issued out of order and that can utilize the issue bandwidth of the processor. To achieve more efficient processing, VLIW fetch packets are preferably filled to capacity to take advantage of code density and issue groups are preferably maximized within each fetch packet.

Detailed Description Text (27):
Preferably, each MAC unit 902 may be capable of one 32.times.32 bit multiply-accumulate function, one 32.times.16 bit multiply-accumulate function, or two 16.times.16 bit multiply-accumulate functions carried out in parallel in one cycle, although they could be designed with other bit size characteristics. Thus, ten MAC units 902 may be available to perform twenty 16.times.16 bit operations concurrently in each cycle. Similarly, preferably each ALU 906 may be capable of one 32.times.32 bit, one 32.times.16 bit, or two 16.times.16 bit arithmetic logic functions carried out in parallel in one cycle. Thus, ten ALUs 906 may be available to perform twenty 16.times.16 bit operations concurrently in each cycle.

Detailed Description Text (28):
Preferably, each load/store unit 908 may be capable of loading or storing either one 32-bit scalar register, a two-element vector where each element is a 32-bit word, or a four-element vector where each element is a 16-bit word. Preferably, each load/store unit may support at least three outstanding loads, i.e., up to four loads in flight in parallel.

Detailed Description Text (31):
The hybrid programming model embodied in a preferred embodiment of the present invention is a flexible model in that it can configure its functional units as needed. For example, a vector MAC instruction can use two MAC units in a single cycle to do a 64-bit vector MAC. On the other hand, in the VLIW mode, four scalar instructions can be issued in parallel and packed into one instruction packet to perform, for example, one MAC, one shift, one load, and one branch in a single cycle. This flexibility provides superior performance on VLIW model operations such as fast fourier transforms ("FFTs") as well SIMD model operations, such as MPEG, FIRs and image processing algorithms.